```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**NTOBLDXAB

```
NN      NN  TTTTTTTTTT    000000    BBBBBBBB    LL          DDDDDDDD    XX      XX    AAAAAA    BBBBBBBB
NN      NN  TTTTTTTTTT    000000    BBBBBBBB    LL          DDDDDDDD    XX      XX    AAAAAA    BBBBBBBB
NN      NN      TT      00      00  BB      BB  LL          DD      DD  XX      XX    AA    AA  BB      BB
NN      NN      TT      00      00  BB      BB  LL          DD      DD  XX      XX    AA    AA  BB      BB
NNNN    NN      TT      00    0000  BB      BB  LL          DD      DD    XX  XX      AA    AA  BB      BB
NNNN    NN      TT      00    0000  BB      BB  LL          DD      DD    XX  XX      AA    AA  BB      BB
NN  NN  NN      TT      00  00  00  BBBBBBBB    LL          DD      DD      XX        AA    AA  BBBBBBBB
NN  NN  NN      TT      00  00  00  BBBBBBBB    LL          DD      DD      XX        AA    AA  BBBBBBBB
NN    NNNN      TT      0000    00  BB      BB  LL          DD      DD    XX  XX      AAAAAAAAAA  BB      BB
NN    NNNN      TT      0000    00  BB      BB  LL          DD      DD    XX  XX      AAAAAAAAAA  BB      BB
NN      NN      TT      00      00  BB      BB  LL          DD      DD  XX      XX    AA    AA  BB      BB    ....
NN      NN      TT      00      00  BB      BB  LL          DD      DD  XX      XX    AA    AA  BB      BB    ....
NN      NN      TT        000000    BBBBBBBB    LLLLLLLLLL  DDDDDDDD    XX      XX    AA    AA  BBBBBBBB    ....
NN      NN      TT        000000    BBBBBBBB    LLLLLLLLLL  DDDDDDDD    XX      XX    AA    AA  BBBBBBBB    ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

(2) 62 DECLARATIONS
(3) 103 NT$ENCODE_KEY
(4) 217 NT$ENCODE_ALL
(5) 318 NT$ENCODE_TIM_D
(6) 451 NT$ENCODE_TIM_R
(7) 525 NT$ENCODE_PRO

```
0000     1          $BEGIN  NTOBLDXAB,000,NF$NETWORK,<BUILD DAP XAB MESSAGES>
0000     2
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; Facility: RMS
0000    31 ;
0000    32 ; Abstract:
0000    33 ;
0000    34 ;       This module builds the DAP Extended Attributes messages.
0000    35 ;
0000    36 ; Environment: VAX/VMS, executive mode
0000    37 ;
0000    38 ; Author: James A. Krycka,      Creation Date:  24-MAY-1979
0000    39 ;
0000    40 ; Modified By:
0000    41 ;
0000    42 ;       V03-006 JAK0145         J A Krycka      12-APR-1984
0000    43 ;               Track changes in DAP message building algorithm.
0000    44 ;
0000    45 ;       V03-005 JAK0132         J A Krycka      17-FEB-1984
0000    46 ;               Always include a menu field in the DAP Protection message.
0000    47 ;
0000    48 ;       V03-004 JAK0124         J A Krycka      06-SEP-1983
0000    49 ;               Make corresponding source code change for VMS V3.5 patch in
0000    50 ;               support of VAXELAN.
0000    51 ;
0000    52 ;       V03-003 KRM0063         K Malik         21-Oct-1982
0000    53 ;               Fix bug in NT$ENCODE_KEY which causes an access violation
0000    54 ;               when KNM field is present.
0000    55 ;
0000    56 ;       V03-002 JAK0101         J A Krycka      09-OCT-1982
0000    57 ;               Build date and time strings with a leading zero instead of
```

```
                        58 ;
                        59 ;
                        60 ;--
```

a leading space to conform to the DAP specification.

```
                     0000     62          .SBTTL  DECLARATIONS
                     0000     63
                     0000     64 ;
                     0000     65 ; Include Files:
                     0000     66 ;
                     0000     67
                     0000     68          $DAPPLGDEF                      ; Define DAP prologue symbols
                     0000     69          $DAPHDRDEF                      ; Define DAP message header
                     0000     70          $DAPATTDEF                      ; Define DAP Attributes message
                     0000     71          $DAPKEYDEF                      ; Define DAP Key Definition message
                     0000     72          $DAPALLDEF                      ; Define DAP Allocation message
                     0000     73          $DAPTIMDEF                      ; Define DAP Date and Time message
                     0000     74          $DAPPRODEF                      ; Define DAP Protection message
                     0000     75          $IFBDEF                         ; Define IFAB symbols
                     0000     76          $NWADEF                         ; Define Network Work Area symbols
                     0000     77          $XABDEF                         ; Define symbols common to all XABs
                     0000     78          $XABALLDEF                      ; Define Allocation XAB symbols
                     0000     79          $XABDATDEF                      ; Define Date and Time XAB symbols
                     0000     80          $XABKEYDEF                      ; Define Key Definition XAB symbols
                     0000     81          $XABPRODEF                      ; Define Protection XAB symbols
                     0000     82          $XABRDTDEF                      ; Define Revision Date/Time XAB symbols
                     0000     83
                     0000     84 ;
                     0000     85 ; Macros:
                     0000     86 ;
                     0000     87 ;       None
                     0000     88 ;
                     0000     89 ; Equated Symbols:
                     0000     90 ;
                     0000     91
                     0000     92          ASSUME  DAP$Q_DCODE_FLG EQ 0
                     0000     93          ASSUME  NWA$Q_FLG EQ 0
                     0000     94
                     0000     95          PARTNER = DAP$W_PARTNER * 8     ; Offset to PARTNER field
                     0000     96
                     0000     97 ;
                     0000     98 ; Own Storage:
                     0000     99 ;
                     0000    100
5D 57 4F 33 21 2C 57 4F 33 21 5B 00' 0000    101 T_UIC:  .ASCIC  \[!30W,!30W]\           ; FAO control string
                       0B    0000
```

```
                     000C   103          .SBTTL  NT$ENCODE_KEY
                     000C   104
                     000C   105 ;++
                     000C   106 ; NT$ENCODE_KEY - builds the DAP Key Definition message.
                     000C   107 ;
                     000C   108 ; Calling Sequence:
                     000C   109 ;
                     000C   110 ;       BSBW    NT$ENCODE_KEY
                     000C   111 ;
                     000C   112 ; Input Parameters:
                     000C   113 ;
                     000C   114 ;       R6      Key Definition XAB address
                     000C   115 ;       R7      NWA (=DAP) address
                     000C   116 ;       R8      FAB address
                     000C   117 ;       R9      IFAB address
                     000C   118 ;       R10     FWA address
                     000C   119 ;       R11     Impure Area address
                     000C   120 ;
                     000C   121 ; Implicit Inputs:
                     000C   122 ;
                     000C   123 ;       User KEYXAB
                     000C   124 ;
                     000C   125 ; Output Parameters:
                     000C   126 ;
                     000C   127 ;       R0-R5   Destroyed
                     000C   128 ;
                     000C   129 ; Implicit Outputs:
                     000C   130 ;
                     000C   131 ;       NWA$Q_BLD
                     000C   132 ;
                     000C   133 ; Completion Codes:
                     000C   134 ;
                     000C   135 ;       None
                     000C   136 ;
                     000C   137 ; Side Effects:
                     000C   138 ;
                     000C   139 ;       None
                     000C   140 ;
                     000C   141 ;--
                     000C   142
                     000C   143 NT$ENCODE_KEY::                         ; Entry point
           50    0A   D0  000C   144          MOVL    #DAP$K_KEY_MSG,R0   ; Get message type value
                FFEE'   30  000F   145          BSBW    NT$BUILD_HEAD       ; Construct message header
     51    000007FF 8F   D0  0012   146          MOVL    #<DAP$M_FLG!-       ; Get key menu value
                     0019   147                    DAP$M_DFL!-         ;
                     0019   148                    DAP$M_IFL!-         ;
                     0019   149                    DAP$M_NSG!-         ;
                     0019   150                    DAP$M_REF!-         ;
                     0019   151                    DAP$M_KNM!-         ;
                     0019   152                    DAP$M_NUL!-         ;
                     0019   153                    DAP$M_IAN!-         ;
                     0019   154                    DAP$M_LAN!-         ;
                     0019   155                    DAP$M_DAN!-         ;
                     0019   156                    DAP$M_DTP!-         ;
                     0019   157                    0>,R1               ;
                FFE4'   30  0019   158          BSBW    NT$CVT_BN4_EXT      ; Store KEYMENU as an extensible field
                     001C   159
```

```
                     001C   160 ;
                     001C   161 ; Include the FLG, DFL, and IFL fields in the message.
                     001C   162 ;
                     001C   163
51   12 A6   9A      001C   164          MOVZBL  XAB$B_FLG(R6),R1          ; Get FLG bits returned by RMS
     52      D4      0020   165          CLRL    R2                        ; Clear corresponding DAP bits
                     0022   166          $MAPBIT XAB$V_DUP,DAP$V_DUP       ; Map DUP bit
                     002A   167          $MAPBIT XAB$V_CHG,DAP$V_CHG       ; Map CHG bit
                     0032   168          $MAPBIT XAB$V_NUL,DAP$V_NUL_CHR   ; Map NUL bit
   85     52      90      003A   169          MOVB    R2,(R5)+                  ; Store FLG as extensible field
85   1C A6   B0      003D   170          MOVW    XAB$W_DFL(R6),(R5)+       ; Store data bucket fill quantity field
85   1A A6   B0      0041   171          MOVW    XAB$W_IFL(R6),(R5)+       ; Store index bucket fill quantity field
                     0045   172
                     0045   173 ;
                     0045   174 ; Include the NSG, POS, and SIZ fields in the message.
                     0045   175 ;
                     0045   176
     50      D4      0045   177          CLRL    R0                        ; Initialize segment counter
     13 A6   91      0047   178          CMPB    XAB$B_DTP(R6),-           ; Branch if the data type of the key
             00      004A   179                  #XAB$C_STG                ;  is string
             04   13 004B   180          BEQL    10$                       ;
             50   D6 004D   181          INCL    R0                        ; It's not string so there can be
             0C   11 004F   182          BRB     30$                       ;  only one segment for the key
52   2E A6   9E      0051   183 10$:     MOVAB   XAB$B_SIZ(R6),R2          ; Get address of SIZ array
             82   95 0055   184 20$:     TSTB    (R2)+                     ; Exit loop on segment size of zero
             04   13 0057   185          BEQL    30$                       ;
F8 50        08   F2 0059   186          AOBLSS  #8,R0,20$                 ; Branch if more segments to examine
   85        50   90 005D   187 30$:     MOVB    R0,(R5)+                  ; Store NSG field
             11   13 0060   188          BEQL    50$                       ; Branch if no segments found
51   1E A6   3E      0062   189          MOVAW   XAB$W_POS(R6),R1          ; Get address of POS array
52   2E A6   9E      0066   190          MOVAB   XAB$B_SIZ(R6),R2          ; Get address of SIZ array
   85   81   B0      006A   191 40$:     MOVW    (R1)+,(R5)+               ; Store next POS field
   85   82   90      006D   192          MOVB    (R2)+,(R5)+               ; Store next SIZ field
     F7 50   F5      0070   193          SOBGTR  R0,40$                    ; Loop if more to go
                     0073   194
                     0073   195 ;
                     0073   196 ; Include the REF, KNM, NUL, IAN, LAN, DAN, and DTP fields in the message.
                     0073   197 ;
                     0073   198
     85   17 A6   90      0073   199 50$:     MOVB    XAB$B_REF(R6),(R5)+       ; Store key of reference field
             85   94 0077   200          CLRB    (R5)+                     ; Assume no key name buffer
     51   38 A6   D0      0079   201          MOVL    XAB$L_KNM(R6),R1          ; Get address of key name buffer
             12   13 007D   202          BEQL    60$                       ; Branch if none specified
61   20      0A A9   0C      007F   203          PROBER  IFB$B_MODE(R9),#32,(R1) ; Probe readability of user buffer
             0B   13 0084   204          BEQL    60$                       ; Branch if not accessible
   FF A5     20   90 0086   205          MOVB    #32,-1(R5)                ; Store KNM as an image field
65   61        20   28 008A   206          MOVC3   #32,(R1),(R5)             ; Copy 32-byte KNM field into message
     55        53   D0 008E   207          MOVL    R3,R5                     ; Update next byte pointer
     85   15 A6   90      0091   208 60$:     MOVB    XAB$B_NUL(R6),(R5)+       ; Store null key character field
     85   08 A6   90      0095   209          MOVB    XAB$B_IAN(R6),(R5)+       ; Store index area number field
     85   09 A6   90      0099   210          MOVB    XAB$B_LAN(R6),(R5)+       ; Store lowest level index area
                     009D   211                                            ;  number field
     85   0A A6   90      009D   212          MOVB    XAB$B_DAN(R6),(R5)+       ; Store data area number field
     85   13 A6   90      00A1   213          MOVB    XAB$B_DTP(R6),(R5)+       ; Store key data type field
       FF58'      30      00A5   214          BSBW    NT$BUILD_TAIL             ; Finish building message
                  05      00A8   215          RSB                               ; Exit
```

```
                                                M 8
                     00A9   217          .SBTTL  NT$ENCODE_ALL
                     00A9   218
                     00A9   219 ;++
                     00A9   220 ; NT$ENCODE_ALL - builds the DAP Allocation message.
                     00A9   221 ;
                     00A9   222 ; Calling Sequence:
                     00A9   223 ;
                     00A9   224 ;       BSBW    NT$ENCODE_ALL
                     00A9   225 ;
                     00A9   226 ; Input Parameters:
                     00A9   227 ;
                     00A9   228 ;       R6      Allocation XAB address
                     00A9   229 ;       R7      NWA (=DAP) address
                     00A9   230 ;       R8      FAB address
                     00A9   231 ;       R9      IFAB address
                     00A9   232 ;       R10     IFAB/FWA address
                     00A9   233 ;       R11     Impure Area address
                     00A9   234 ;
                     00A9   235 ; Implicit Inputs:
                     00A9   236 ;
                     00A9   237 ;       User ALLXAB
                     00A9   238 ;       DAP$V_STM_ONLY
                     00A9   239 ;       DAP$V_VAXVMS, DAP$V_VAXELAN
                     00A9   240 ;
                     00A9   241 ; Output Parameters:
                     00A9   242 ;
                     00A9   243 ;       R0-R5   Destroyed
                     00A9   244 ;
                     00A9   245 ; Implicit Outputs:
                     00A9   246 ;
                     00A9   247 ;       NWA$Q_BLD
                     00A9   248 ;
                     00A9   249 ; Completion Codes:
                     00A9   250 ;
                     00A9   251 ;       None
                     00A9   252 ;
                     00A9   253 ; Side Effects:
                     00A9   254 ;
                     00A9   255 ;       None
                     00A9   256 ;
                     00A9   257 ;--
                     00A9   258
                     00A9   259 NT$ENCODE_ALL::                         ; Entry point
          50   0B   D0  00A9   260          MOVL    #DAP$K_ALL_MSG,R0    ; Get message type value
               FF51'  30  00AC   261          BSBW    NT$BUILD_HEAD        ; Construct message header
51   01E5 8F   3C  00AF   262          MOVZWL  #<DAP$M_VOL!-                ; Get initial allocation menu value
                     00B4   263                    DAP$M_AOP!-           ;
                     00B4   264                    DAP$M_ALQ2!-          ;
                     00B4   265                    DAP$M_AID!-           ;
                     00B4   266                    DAP$M_BKZ!-           ;
                     00B4   267                    DAP$M_DEQ2!-          ;
                     00B4   268                    0>,R1                 ;
03 67   34   E1  00B4   269          BBC     #DAP$V_VAXVMS,(R7),10$       ; Branch if partner is not VAX/VMS
     51   0A   A8  00B8   270          BISW2   #<<DAP$M_ALN>!-              ; Add ALN and LOC fields to menu
                     00BB   271                    <DAP$M_LOC>!-         ;
                     00BB   272                    0>,R1                 ;
               FF42'  30  00BB   273 10$:    BSBW    NT$CVT_BN4_EXT       ; Store ALLMENU as an extensible field
```

```
                        00BE   274
                        00BE   275 ;
                        00BE   276 ; Include the VOL, ALN, and AOP fields in the message.
                        00BE   277 ;
                        00BE   278
85   0A A6      B0      00BE   279          MOVW    XAB$W_VOL(R6),(R5)+     ; Store relative volume number field
04 67   34      E1      00C2   280          BBC     #DAP$V_VAXVMS,(R7),20$  ; Branch if partner is not VAX/VMS
                        00C6   281
                        00C6   282          ASSUME  DAP$K_ANY EQ 0
                        00C6   283          ASSUME  DAP$K_CYL EQ XAB$C_CYL
                        00C6   284          ASSUME  DAP$K_LBN EQ XAB$C_LBN
                        00C6   285          ASSUME  DAP$K_VBN EQ XAB$C_VBN
                        00C6   286
85   09 A6      90      00C6   287          MOVB    XAB$B_ALN(R6),(R5)+     ; Store alignment options field
51   08 A6      9A      00CA   288 20$:      MOVZBL  XAB$B_AOP(R6),R1        ; Get AOP bits returned by RMS
           52    D4      00CE   289          CLRL    R2                      ; Clear corresponding DAP bits
                        00D0   290          $MAPBIT XAB$V_CTG,DAP$V_CTG2    ; Map CTG bit
06 A7      30      B3      00D8   291          BITW    #<<1@<DAP$V_VAXVMS-PARTNER>>!-
                        00DC   292                    <1@<DAP$V_VAXELAN-PARTNER>>!-
                        00DC   293                    0>,DAP$W_PARTNER(R7)    ; Branch if partner is neither VAX/VMS
           18    13      00DC   294          BEQL    30$                     ;  nor VAXELAN
                        00DE   295          $MAPBIT XAB$V_CBT,DAP$V_CBT2    ; Map CBT bit
                        00E6   296          $MAPBIT XAB$V_HRD,DAP$V_HRD     ; Map HRD bit
                        00EE   297          $MAPBIT XAB$V_ONC,DAP$V_ONC     ; Map ONC bit
   51    52    D0      00F6   298 30$:      MOVL    R2,R1                   ; Move data to correct register
      FF04'    30      00F9   299          BSBW    NT$CVT_BN4_EXT          ; Store AOP as an extensible field
                        00FC   300
                        00FC   301 ;
                        00FC   302 ; Include the LOC, ALQ, AID, BKZ, and DEQ fields in the message.
                        00FC   303 ;
                        00FC   304
07 67   34      E1      00FC   305          BBC     #DAP$V_VAXVMS,(R7),40$  ; Branch if partner is not VAX/VMS
51   0C A6      D0      0100   306          MOVL    XAB$L_LOC(R6),R1        ; Get starting location value
      FEF9'     30      0104   307          BSBW    NT$CVT_BN4_IMG          ; Store LOC as an image field
51   10 A6      D0      0107   308 40$:      MOVL    XAB$L_ALQ(R6),R1        ; Get allocation quantity value
02 67   32      E1      010B   309          BBC     #DAP$V_STM_ONLY,(R7),50$; Branch if not a 'stream-only' machine
           51    D4      010F   310          CLRL    R1                      ; Send ALQ value of zero
      FEEC'     30      0111   311 50$:      BSBW    NT$CVT_BN4_IMG          ; Store ALQ as an image field
85   17 A6      90      0114   312          MOVB    XAB$B_AID(R6),(R5)+     ; Store area identification field
85   16 A6      90      0118   313          MOVB    XAB$B_BKZ(R6),(R5)+     ; Store bucket size field
85   14 A6      B0      011C   314          MOVW    XAB$W_DEQ(R6),(R5)+     ; Store default extension quantity field
      FEDD'     30      0120   315          BSBW    NT$BUILD_TAIL           ; Finish building message
                05      0123   316          RSB                             ; Exit
```

```
                     0124    318          .SBTTL  NT$ENCODE_TIM_D
                     0124    319
                     0124    320 ;++
                     0124    321 ; NT$ENCODE_TIM_D - builds the DAP Date and Time message using the Date and Time
                     0124    322 ;       XAB as input.
                     0124    323 ;
                     0124    324 ; Calling Sequence:
                     0124    325 ;
                     0124    326 ;       BSBW    NT$ENCODE_TIM_D
                     0124    327 ;
                     0124    328 ; Input Parameters:
                     0124    329 ;
                     0124    330 ;       R6      Date and Time XAB address
                     0124    331 ;       R7      NWA (=DAP) address
                     0124    332 ;       R8      FAB address
                     0124    333 ;       R9      IFAB address
                     0124    334 ;       R10     IFAB/FWA address
                     0124    335 ;       R11     Impure Area address
                     0124    336 ;
                     0124    337 ; Implicit Inputs:
                     0124    338 ;
                     0124    339 ;       User DATXAB
                     0124    340 ;       DAP$V_GEQ_V60
                     0124    341 ;
                     0124    342 ; Output Parameters:
                     0124    343 ;
                     0124    344 ;       R0-R5   Destroyed
                     0124    345 ;
                     0124    346 ; Implicit Outputs:
                     0124    347 ;
                     0124    348 ;       NWA$Q_BLD
                     0124    349 ;
                     0124    350 ; Completion Codes:
                     0124    351 ;
                     0124    352 ;       None
                     0124    353 ;
                     0124    354 ; Side Effects:
                     0124    355 ;
                     0124    356 ;       None
                     0124    357 ;
                     0124    358 ;--
                     0124    359
                     0124    360 NT$ENCODE_TIM_D::                       ; Entry point
 50   0D   D0        0124    361         MOVL    #DAP$K_TIM_MSG,R0      ; Get message type value
    FED6'  30        0127    362         BSBW    NT$BUILD_HEAD          ; Construct message header
                     012A    363
                     012A    364 ;
                     012A    365 ; Construct value for date and time menu field.
                     012A    366 ; Send only time fields that have a non-zero 64-bit time value, as zero means
                     012A    367 ; the current date and time, not 17-NOV-1858! (Actually only the upper 32-bits
                     012A    368 ; will be tested for zero.)
                     012A    369 ;
                     012A    370
                     012A    371         ASSUME  DAP$V_CDT LT 7
                     012A    372         ASSUME  DAP$V_RDT LT 7
                     012A    373         ASSUME  DAP$V_EDT LT 7
                     012A    374         ASSUME  DAP$V_RVN LT 7
```

```
                     012A   375          ASSUME  DAP$V_BDT LT 7
                     012A   376
          54   D4    012A   377          CLRL    R4                      ; Initialize time menu field
       18 A6   D5    012C   378          TSTL    XAB$Q_CDT+4(R6)         ; Branch if creation date and time
          03   13    012F   379          BEQL    10$                     ;  is zero
       54 01   88    0131   380          BISB2   #DAP$M_CDT,R4           ; Otherwise, send field
       10 A6   D5    0134   381 10$:     TSTL    XAB$Q_RDT+4(R6)         ; Branch if revision date and time
          03   13    0137   382          BEQL    20$                     ;  is zero
       54 02   88    0139   383          BISB2   #DAP$M_RDT,R4           ; Otherwise, send field
       20 A6   D5    013C   384 20$:     TSTL    XAB$Q_EDT+4(R6)         ; Branch if expiration date and time
          03   13    013F   385          BEQL    30$                     ;  is zero
       54 04   88    0141   386          BISB2   #DAP$M_EDT,R4           ; Otherwise, send field
    0E 67 25   E1    0144   387 30$:     BBC     #DAP$V_GEQ_V60,(R7),40$ ; Branch if partner uses DAP before V6.0
       01 A6   91    0148   388          CMPB    XAB$B_BLN(R6),-         ; Branch if length of XAB is too small
          2C             014B   389                  #XAB$C_DATLEN           ;  to contain BDT field (i.e., it's a
          08   1F    014C   390          BLSSU   40$                     ;  V2 length XAB)
       28 A6   D5    014E   391          TSTL    XAB$Q_BDT+4(R6)         ; Branch if backup date and time
          03   13    0151   392          BEQL    40$                     ;  is zero
       54 10   88    0153   393          BISB2   #DAP$M_BDT,R4           ; Otherwise, send field
       54 08   88    0156   394 40$:     BISB2   #DAP$M_RVN,R4           ; Send revision number field
       85 54   90    0159   395          MOVB    R4,(R5)+                ; Store TIMENU as an extensible field
                     015C   396
                     015C   397 ;
                     015C   398 ; Now process each field.
                     015C   399 ;
                     015C   400
    06 54 00   E1    015C   401          BBC     #DAP$V_CDT,R4,50$       ; Branch if CDT is not to be included
    50 14 A6   7E    0160   402          MOVAQ   XAB$Q_CDT(R6),R0        ; Get address of 64-bit value for
                     0164   403                                          ;  creation date and time
          26   10    0164   404          BSBB    CONVERT_TIME            ; Store CDT as an image field
    06 54 01   E1    0166   405 50$:     BBC     #DAP$V_RDT,R4,60$       ; Branch if RDT is not to be included
    50 0C A6   7E    016A   406          MOVAQ   XAB$Q_RDT(R6),R0        ; Get address of 64-bit value for
                     016E   407                                          ;  revision date and time
          1C   10    016E   408          BSBB    CONVERT_TIME            ; Store RDT as an image field
    06 54 02   E1    0170   409 60$:     BBC     #DAP$V_EDT,R4,70$       ; Branch if EDT is not to be included
    50 1C A6   7E    0174   410          MOVAQ   XAB$Q_EDT(R6),R0        ; Get address of 64-bit value for
                     0178   411                                          ;  expiration date and time
          12   10    0178   412          BSBB    CONVERT_TIME            ; Store EDT as an image field
    85 08 A6   B0    017A   413 70$:     MOVW    XAB$W_RVN(R6),(R5)+     ; Store revision number field
    06 54 04   E1    017E   414          BBC     #DAP$V_BDT,R4,80$       ; Branch if BDT is not to be included
    50 24 A6   7E    0182   415          MOVAQ   XAB$Q_BDT(R6),R0        ; Get address of 64-bit value for
                     0186   416                                          ;  backup date and time
          04   10    0186   417          BSBB    CONVERT_TIME            ; Store BDT as an image field
       FE75' 30      0188   418 80$:     BSBW    NT$BUILD_TAIL           ; Finish building message
             05      018B   419          RSB                             ; Exit
                     018C   420
                     018C   421 ;+
                     018C   422 ; This routine converts a time value in 64-bit binary format to an ASCII string.
                     018C   423 ; Then it stores the string as an 18-byte fixed length field in the DAP message
                     018C   424 ; with the first two digits of the year removed (per DAP spec).
                     018C   425 ;-
                     018C   426
                     018C   427 CONVERT_TIME:                            ; Entry point
       5E   20   C2  018C   428          SUBL2   #<20+12>,SP             ; Allocate space from the stack
       52   5E   D0  018F   429          MOVL    SP,R2                   ; Save address of work area
    14 A2   14   D0  0192   430          MOVL    #20,20(R2)              ; Form descriptor of buffer to receive
    18 A2   5E   D0  0196   431          MOVL    SP,24(R2)               ;  ASCII time string
```

```
                     019A   432          $ASCTIM_S-                      ; Convert binary time to ASCII time
                     019A   433                   TIMLEN=28(R2)-         ;  Address of word to receive string size
                     019A   434                   TIMBUF=20(R2)-         ;  Address of descriptor for buffer
                     019A   435                   TIMADR=(R0)-           ;  Address of 64-bit time value
                     019A   436                   CVTFLG=#0              ;  Flag set to request date and time
                     01AB   437                                          ; Assume success; ignore failure
          62 20 91   01AB   438          CMPB     #^A\ \,(R2)            ; Convert leading space to zero in
             03 12   01AE   439          BNEQ     10$                    ;  day-of-month field to conform to
          62 30 90   01B0   440          MOVB     #^A\0\,(R2)            ;  the DAP V6.0 specification
                     01B3   441                                          ; Store time field omitting the two
                     01B3   442                                          ;  century digits
             10 BB   01B3   443 10$:     PUSHR    #^M<R4>                ; Save time menu mask
       65 62 07 28   01B5   444          MOVC3    #7,(R2),(R5)           ; Copy bytes 1-7 of input string
 63    02 A1 0B 28   01B9   445          MOVC3    #11,2(R1),(R3)         ; Copy bytes 9-20 of input string
             10 BA   01BE   446          POPR     #^M<R4>                ; Restore time menu mask
          55 53 D0   01C0   447          MOVL     R3,R5                  ; Update next byte pointer
          5E 20 C0   01C3   448          ADDL2    #<20+12>,SP            ; Deallocate space from the stack
             05      01C6   449          RSB                             ; Exit
```

```
                        01C7   451          .SBTTL  NT$ENCODE_TIM_R
                        01C7   452
                        01C7   453 ;++
                        01C7   454 ; NT$ENCODE_TIM - builds the DAP Date and Time message using the Revision
                        01C7   455 ;       Date and Time XAB as input.
                        01C7   456 ;
                        01C7   457 ; Calling Sequence:
                        01C7   458 ;
                        01C7   459 ;       BSBW    NT$ENCODE_TIM_R
                        01C7   460 ;
                        01C7   461 ; Input Parameters:
                        01C7   462 ;
                        01C7   463 ;       R6      Revision Date and Time XAB address
                        01C7   464 ;       R7      NWA (=DAP) address
                        01C7   465 ;       R8      FAB address
                        01C7   466 ;       R9      IFAB address
                        01C7   467 ;       R10     IFAB/FWA address
                        01C7   468 ;       R11     Impure Area address
                        01C7   469 ;
                        01C7   470 ; Implicit Inputs:
                        01C7   471 ;
                        01C7   472 ;       User RDTXAB
                        01C7   473 ;
                        01C7   474 ; Output Parameters:
                        01C7   475 ;
                        01C7   476 ;       R0-R5   Destroyed
                        01C7   477 ;
                        01C7   478 ; Implicit Outputs:
                        01C7   479 ;
                        01C7   480 ;       NWA$Q_BLD
                        01C7   481 ;
                        01C7   482 ; Completion Codes:
                        01C7   483 ;
                        01C7   484 ;       None
                        01C7   485 ;
                        01C7   486 ; Side Effects:
                        01C7   487 ;
                        01C7   488 ;       None
                        01C7   489 ;
                        01C7   490 ;--
                        01C7   491
                        01C7   492 NT$ENCODE_TIM_R::                            ; Entry point
50   0D   D0   01C7   493          MOVL    #DAP$K_TIM_MSG,R0                   ; Get message type value
  FE33'  30   01CA   494          BSBW    NT$BUILD_HEAD                       ; Construct message header
                        01CD   495
                        01CD   496 ;
                        01CD   497 ; Construct value for date and time menu field.
                        01CD   498 ; Send only time fields that have a non-zero 64-bit time value, as zero means
                        01CD   499 ; the current date and time, not 17-NOV-1858! (Actually only the upper 32-bits
                        01CD   500 ; will be tested for zero.)
                        01CD   501 ;
                        01CD   502
                        01CD   503          ASSUME  DAP$V_RDT LT 7
                        01CD   504          ASSUME  DAP$V_RVN LT 7
                        01CD   505
   54   D4   01CD   506          CLRL    R4                                  ; Initialize time menu field
10 A6   D5   01CF   507          TSTL    XAB$Q_RDT+4(R6)                     ; Branch if revision date and time
```

```
                03   13  01D2   508                 BEQL    10$                             ;  is zero
             54 02   88  01D4   509                 BISB2   #DAP$M_RDT,R4                   ; Otherwise, send field
             54 08   88  01D7   510 10$:            BISB2   #DAP$M_RVN,R4                   ; Send revision number field
             85 54   90  01DA   511                 MOVB    R4,(R5)+                        ; Store TIMENU as an extensible field
                         01DD   512
                         01DD   513 ;
                         01DD   514 ; Now process each field.
                         01DD   515 ;
                         01DD   516
          06 54   01     E1  01DD   517                 BBC     #DAP$V_RDT,R4,30$               ; Branch if RDT is not to be included
          50   0C A6     7E  01E1   518                 MOVAQ   XAB$Q_RDT(R6),R0                ; Get address of 64-bit value for
                         01E5   519                                                         ;  revision date and time
                   A5    10  01E5   520                 BSBB    CONVERT_TIME                    ; Store RDT as an image field
          85   08 A6     B0  01E7   521 30$:            MOVW    XAB$W_RVN(R6),(R5)+             ; Store revision number field
                FE12'    30  01EB   522                 BSBW    NT$BUILD_TAIL                   ; Finish building message
                         05  01EE   523                 RSB                                     ; Exit
```

```
                     01EF    525          .SBTTL  NT$ENCODE_PRO
                     01EF    526
                     01EF    527 ;++
                     01EF    528 ; NT$ENCODE_PRO - builds the DAP Protection message.
                     01EF    529 ;
                     01EF    530 ; Calling Sequence:
                     01EF    531 ;
                     01EF    532 ;       BSBW    NT$ENCODE_PRO
                     01EF    533 ;
                     01EF    534 ; Input Parameters:
                     01EF    535 ;
                     01EF    536 ;       R6      Protection XAB address
                     01EF    537 ;       R7      NWA (=DAP) address
                     01EF    538 ;       R8      FAB address
                     01EF    539 ;       R9      IFAB address
                     01EF    540 ;       R10     IFAB/FWA address
                     01EF    541 ;       R11     Impure Area address
                     01EF    542 ;
                     01EF    543 ; Implicit Inputs:
                     01EF    544 ;
                     01EF    545 ;       User PROXAB
                     01EF    546 ;
                     01EF    547 ; Output Parameters:
                     01EF    548 ;
                     01EF    549 ;       R0-R5   Destroyed
                     01EF    550 ;
                     01EF    551 ; Implicit Outputs:
                     01EF    552 ;
                     01EF    553 ;       NWA$Q_BLD
                     01EF    554 ;
                     01EF    555 ; Completion Codes:
                     01EF    556 ;
                     01EF    557 ;       None
                     01EF    558 ;
                     01EF    559 ; Side Effects:
                     01EF    560 ;
                     01EF    561 ;       None
                     01EF    562 ;
                     01EF    563 ;--
                     01EF    564
                     01EF    565 NT$ENCODE_PRO::                             ; Entry point
0800 8F     BB       01EF    566          PUSHR   #^M<R11>                   ; Save register
50     0E   D0       01F3    567          MOVL    #DAP$K_PRO_MSG,R0          ; Get message type value
    FE07'   30       01F6    568          BSBW    NT$BUILD_HEAD              ; Construct message header
                     01F9    569
                     01F9    570          ASSUME  DAP$V_OWNER LT 7
                     01F9    571          ASSUME  DAP$V_PROSYS LT 7
                     01F9    572          ASSUME  DAP$V_PROOWN LT 7
                     01F9    573          ASSUME  DAP$V_PROGRP LT 7
                     01F9    574          ASSUME  DAP$V_PROWLD LT 7
                     01F9    575
       5B    D4      01F9    576          CLRL    R11                        ; Initialize temp PROMENU
     0C A6   D5      01FB    577          TSTL    XAB$L_UIC(R6)              ; Is UIC value [0,0]?
          03   13    01FE    578          BEQL    10$                        ; Branch if yes
    5B    01   88    0200    579          BISB2   #DAP$M_OWNER,R11           ; Set OWNER bit in temp PROMENU
08 A6  FFFF 8F     B1 0203    580 10$:     CMPW    #-1,XAB$W_PRO(R6)          ; Are the 4 protection fields defaulted?
          03   13    0209    581          BEQL    20$                        ; Branch if yes
```

```
                5B   1E   88   020B   582          BISB2   #<DAP$M_PROSYS!-          ; Set temp PROMENU bits
                                 020E   583                    DAP$M_PROOWN!-          ;
                                 020E   584                    DAP$M_PROGRP!-          ;
                                 020E   585                    DAP$M_PROWLD>,R11       ;
                85   5B   90   020E   586 20$:     MOVB    R11,(R5)+                 ; Store PROMENU as an extensible field
                     7D   13   0211   587         BEQL    70$                       ; Branch if no more fields to send
                               0213   588
                               0213   589 ;
                               0213   590 ; Include the OWNER field in the message.
                               0213   591 ;
                               0213   592
              4E 5B   00   E1   0213   593         BBC     #DAP$V_OWNER,R11,60$      ; Branch if no OWNER field
                 5E   24   C2   0217   594         SUBL2   #<16+8+8+4>,SP            ; Allocate space from the stack
                 52   5E   D0   021A   595         MOVL    SP,R2                     ; Save address of work area
                10 A2   10   D0   021D   596         MOVL    #16,16(R2)                ; Form descriptor of buffer to receive
                14 A2   5E   D0   0221   597         MOVL    SP,20(R2)                 ;  ASCII string
          18 A2    FDD7   CF   9A   0225   598         MOVZBL  W^T_UIC,24(R2)            ; From descriptor of FAO control
          1C A2    FDD2   CF   9E   022B   599         MOVAB   W^T_UIC+1,28(R2)          ;  string
                50   0E A6   3C   0231   600         MOVZWL  XAB$W_GRP(R6),R0          ; Get group UIC value
                51   0C A6   3C   0235   601         MOVZWL  XAB$W_MBM(R6),R1          ; Get member UIC value
                               0239   602         $FAO_S-                           ; Format the UIC string
                               0239   603                 CTRSTR=24(R2)-            ;  FAO control string
                               0239   604                 OUTLEN=32(R2)-            ;  Address to receive string length
                               0239   605                 OUTBUF=16(R2)-            ;  Address of buffer descriptor
                               0239   606                 P1=R0-                    ;  Group number of file owner
                               0239   607                 P2=R1                     ;  Member number of file owner
                   04   50   E8   024D   608         BLBS    R0,40$                    ; Branch on success
                        85   94   0250   609         CLRB    (R5)+                     ; Send null OWNER field
                        0E   11   0252   610         BRB     50$                       ;
                   50   20 A2   3C   0254   611 40$:     MOVZWL  32(R2),R0         ; Get length of returned string
                             85   50   90   0258   612         MOVB    R0,(R5)+          ; Store OWNER as an image field
                          65   62   50   28   025B   613         MOVC3   R0,(R2),(R5)      ; Copy owner string to message
                               55   53   D0   025F   614         MOVL    R3,R5             ; Update next byte pointer
                               5E   24   C0   0262   615 50$:     ADDL2   #<16+8+8+4>,SP    ; Deallocate space from the stack
                                         0265   616
                                         0265   617 ;
                                         0265   618 ; Construct the four protection fields: PROSYS, PROOWN, PROGRP, and PROWLD.
                                         0265   619 ;
                                         0265   620
        50    5B   04   01   EF   0265   621 60$:     EXTZV   #DAP$V_PROSYS,#4,R11,R0 ; Get the protection bits to check
                               24   13   026A   622         BEQL    70$               ; Branch if they're all defaulted
                                         026C   623
                                         026C   624         ASSUME  DAP$V_RED_ACC EQ XAB$V_NOREAD
                                         026C   625         ASSUME  DAP$V_WRT_ACC EQ XAB$V_NOWRITE
                                         026C   626         ASSUME  DAP$V_EXE_ACC EQ XAB$V_NOEXE
                                         026C   627         ASSUME  DAP$V_DLT_ACC EQ XAB$V_NODEL
                                         026C   628
                                         026C   629         ASSUME  DAP$V_RED_ACC LT 7
                                         026C   630         ASSUME  DAP$V_WRT_ACC LT 7
                                         026C   631         ASSUME  DAP$V_EXE_ACC LT 7
                                         026C   632         ASSUME  DAP$V_DLT_ACC LT 7
                                         026C   633
                   50   08 A6   3C   026C   634         MOVZWL  XAB$W_PRO(R6),R0          ; Get protection value
        51    50   04   00   EF   0270   635         EXTZV   #XAB$V_SYS,#4,R0,R1       ; Store system protection field
                        85   51   90   0275   636         MOVB    R1,(R5)+          ;  as an extensible field
        51    50   04   04   EF   0278   637         EXTZV   #XAB$V_OWN,#4,R0,R1       ; Store owner protection field
                        85   51   90   027D   638         MOVB    R1,(R5)+          ;  as an extensible field
```

```
                51   50   04   08   EF  0280   639          EXTZV   #XAB$V_GRP,#4,R0,R1     ; Store group protection field
                                85   51   90  0285   640          MOVB    R1,(R5)+                ;  as an extensible field
                51   50   04   0C   EF  0288   641          EXTZV   #XAB$V_WLD,#4,R0,R1     ; Store world protection field
                                85   51   90  028D   642          MOVB    R1,(R5)+                ;  as an extensible field
                          FD6D'      30  0290   643 70$:      BSBW    NT$BUILD_TAIL           ; Finish building message
                     0800 8F         BA  0293   644          POPR    #^M<R11>                ; Restore register
                                     05  0297   645          RSB                             ; Exit
                                         0298   646
                                         0298   647          .END                            ; End of module
```

```
$$.PSECT_EP                     =  00000000
$$RMSTEST                       =  0000001A
$$RMS_PBUGCHK                   =  00000010
$$RMS_TBUGCHK                   =  00000008
$$RMS_UMODE                     =  00000004
$ST2                            =  00000005
CONVERT_TIME                       0000018C R      01
DAP$B_AID                          00000050
DAP$B_ALN                          00000044
DAP$B_AOP                          00000045
DAP$B_BITCNT                       00000035
DAP$B_BKS                          00000050
DAP$B_BKZ                          00000051
DAP$B_BSZ                          00000052
DAP$B_DAN                          00000070
DAP$B_DATATYPE                     00000044
DAP$B_DBS                          0000007C
DAP$B_DCODE_FID                    00000019
DAP$B_DCODE_MAC                    0000001B
DAP$B_DCODE_MSG                    0000001A
DAP$B_DTP                          00000071
DAP$B_FLAGS                        00000031
DAP$B_FLG                          00000048
DAP$B_FSZ                          00000051
DAP$B_IAN                          0000006E
DAP$B_IBS                          0000007D
DAP$B_LAN                          0000006F
DAP$B_LEN256                       00000034
DAP$B_LENGTH                       00000033
DAP$B_LVL                          0000007E
DAP$B_NSG                          00000049
DAP$B_NUL                          0000006D
DAP$B_ORG                          00000045
DAP$B_RAT                          00000047
DAP$B_REF                          0000006C
DAP$B_RFM                          00000046
DAP$B_SIZ                          0000005C
DAP$B_SIZ_TMP                      0000004A
DAP$B_STREAMID                     00000032
DAP$B_TKS                          0000007F
DAP$B_TYPE                         00000030
DAP$B_X_FIELD                      00000024
DAP$C_BCN                          000000C0
DAP$K_ALL_MSG                   =  00000008
DAP$K_ANY                       =  00000000
DAP$K_BLN                          000000C0
DAP$K_CYL                       =  00000001
DAP$K_FIX                       =  00000001
DAP$K_KEY_MSG                   =  0000000A
DAP$K_LBN                       =  00000002
DAP$K_PRO_MSG                   =  0000000E
DAP$K_SEQ                       =  00000000
DAP$K_STG                       =  00000000
DAP$K_TIM_MSG                   =  0000000D
DAP$K_VBN                       =  00000003
DAP$L_ALQ1                         0000004C
DAP$L_ALQ2                         0000004C
DAP$L_ATTMENU                      00000040
DAP$L_CMWA                         00000030
DAP$L_CRC_RSLT                     00000020
DAP$L_DCODE_STS                    00000018
DAP$L_DEV                          00000068
DAP$L_DVB                          00000078
DAP$L_EBK                          00000078
DAP$L_FOP1                         00000064
DAP$L_HBK                          00000074
DAP$L_KEYMENU                      00000040
DAP$L_LOC                          00000048
DAP$L_MRN                          00000058
DAP$L_MSG_MASK                     0000001C
DAP$L_RVB                          00000074
DAP$L_SBN                          0000007C
DAP$L_SSPWA                        00000080
DAP$L_TEMP                         00000090
DAP$M_AID                       =  00000040
DAP$M_ALN                       =  00000002
DAP$M_ALQ2                      =  00000020
DAP$M_AOP                       =  00000004
DAP$M_BDT                       =  00000010
DAP$M_BITCNT                    =  00000008
DAP$M_BKZ                       =  00000080
DAP$M_CDT                       =  00000001
DAP$M_CMPFMT                    =  00000008
DAP$M_DAN                       =  00000200
DAP$M_DEQ2                      =  00000100
DAP$M_DFL                       =  00000002
DAP$M_DMO                       =  00002000
DAP$M_DTP                       =  00000400
DAP$M_EDT                       =  00000004
DAP$M_EMBEDDED                  =  00000010
DAP$M_FLG                       =  00000001
DAP$M_IAN                       =  00000080
DAP$M_IFL                       =  00000004
DAP$M_IMAGE                     =  00000002
DAP$M_KNM                       =  00000020
DAP$M_LAN                       =  00000100
DAP$M_LOC                       =  00000008
DAP$M_LSA                       =  00000040
DAP$M_MACY11                    =  00000080
DAP$M_NSG                       =  00000008
DAP$M_NUL                       =  00000040
DAP$M_OWNER                     =  00000001
DAP$M_PROGRP                    =  00000008
DAP$M_PROOWN                    =  00000004
DAP$M_PROSYS                    =  00000002
DAP$M_PROWLD                    =  00000010
DAP$M_RDT                       =  00000002
DAP$M_REF                       =  00000010
DAP$M_RVN                       =  00000008
DAP$M_SEGMENT                   =  00000040
DAP$M_TMP1$                     =  0000FE00
DAP$M_TMP2$                     =  0000FE00
DAP$M_TMP3$                     =  00020000
DAP$M_TMP4$                     =  01000000
```

```
DAP$M_TMP5$                     = F0000000
DAP$M_VOL                       = 00000001
DAP$M_ZERO                      = 00000080
DAP$Q_ADT                         00000070
DAP$Q_BDT                         00000060
DAP$Q_CDT                         00000048
DAP$Q_DCODE_FLG                   00000000
DAP$Q_EDT                         00000058
DAP$Q_KNM                         00000064
DAP$Q_MSG_BUF1                    00000008
DAP$Q_MSG_BUF2                    00000010
DAP$Q_OWNER                       00000048
DAP$Q_PDT                         00000068
DAP$Q_RDT                         00000050
DAP$Q_RUNSYS                      0000005C
DAP$Q_SYSPEC                      00000038
DAP$V_BDT                       = 00000004
DAP$V_CBT2                      = 00000002
DAP$V_CDT                       = 00000000
DAP$V_CHG                       = 00000001
DAP$V_CTG2                      = 00000001
DAP$V_DLT_ACC                   = 00000003
DAP$V_DUP                       = 00000000
DAP$V_EDT                       = 00000002
DAP$V_EXE_ACC                   = 00000002
DAP$V_GEQ_V60                   = 00000025
DAP$V_HRD                       = 00000000
DAP$V_NUL_CHR                   = 00000002
DAP$V_ONC                       = 00000003
DAP$V_OWNER                     = 00000000
DAP$V_PROGRP                    = 00000003
DAP$V_PROOWN                    = 00000002
DAP$V_PROSYS                    = 00000001
DAP$V_PROWLD                    = 00000004
DAP$V_RDT                       = 00000001
DAP$V_RED_ACC                   = 00000000
DAP$V_RVN                       = 00000003
DAP$V_STM_ONLY                  = 00000032
DAP$V_VAXELAN                   = 00000035
DAP$V_VAXVMS                    = 00000034
DAP$V_WRT_ACC                   = 00000001
DAP$W_ALLMENU                     00000040
DAP$W_BLS                         00000048
DAP$W_DEQ1                        00000054
DAP$W_DEQ2                        00000052
DAP$W_DFL                         00000044
DAP$W_FFB                         00000072
DAP$W_IFL                         00000046
DAP$W_LRL                         00000070
DAP$W_MRL                         00000072
DAP$W_MRS                         0000004A
DAP$W_PARTNER                     00000006
DAP$W_POS                         0000004C
DAP$W_POS_TMP                     0000004A
DAP$W_PROGRP                      00000054
DAP$W_PROMENU                     00000040
DAP$W_PROOWN                      00000052
DAP$W_PROSYS                      00000050
DAP$W_PROWLD                      00000056
DAP$W_RVN                         00000042
DAP$W_TIMENU                      00000040
DAP$W_VERSION                     00000004
DAP$W_VOL                         00000042
IFB$B_MODE                      = 0000000A
NT$BUILD_HEAD                     ********    X   01
NT$BUILD_TAIL                     ********    X   01
NT$CVT_BN4_EXT                    ********    X   01
NT$CVT_BN4_IMG                    ********    X   01
NT$ENCODE_ALL                     000000A9 RG     01
NT$ENCODE_KEY                     0000000C RG     01
NT$ENCODE_PRO                     000001EF RG     01
NT$ENCODE_TIM_D                   00000124 RG     01
NT$ENCODE_TIM_R                   000001C7 RG     01
NWA$B_ALLXABCNT                   0000011C
NWA$B_DAP_RAC                     000000C9
NWA$B_FILESYS                     000000C5
NWA$B_KEYXABCNT                   0000011D
NWA$B_NETSTRSIZ                   0000016F
NWA$B_NODBUFSIZ                   00000168
NWA$B_ORG                         000000C6
NWA$B_OSTYPE                      000000C4
NWA$B_RFM                         000000C7
NWA$B_RMS_RAC                     000000C8
NWA$C_BLN                         00000800
NWA$K_BLN                         00000800
NWA$L_ALLXABADR                   00000100
NWA$L_DATXABADR                   00000104
NWA$L_DEV                         000000C0
NWA$L_FHCXABADR                   00000108
NWA$L_KEYXABADR                   0000010C
NWA$L_MSG_MASK                    000000D4
NWA$L_PROXABADR                   00000110
NWA$L_RDTXABADR                   00000114
NWA$L_SAVE_FLGS                   00000128
NWA$L_SUMXABADR                   00000118
NWA$L_THREAD                      000000FC
NWA$L_XLTATTR                     00000238
NWA$L_XLTBUFFLG                   0000022C
NWA$L_XLTCNT                      00000228
NWA$L_XLTMAXINDX                  00000234
NWA$L_XLTSIZ                      00000230
NWA$Q_ACS                         00000244
NWA$Q_BIGBUF                      00000170
NWA$Q_BLD                         000000F0
NWA$Q_FLG                         00000000
NWA$Q_INODE                       0000025C
NWA$Q_IOSB                        000000D8
NWA$Q_LNODE                       00000160
NWA$Q_LOGNAME                     0000023C
NWA$Q_NCB                         00000264
NWA$Q_RCV                         000000E0
NWA$Q_SAVE_DESC                   00000120
NWA$Q_XLTBUF1                     0000024C
NWA$Q_XLTBUF2                     00000254
```

```
NWA$Q_XMT                       000000E8
NWA$T_ACSBUF                    0000026C
NWA$T_AUXBUF                    000005E0
NWA$T_DAP                       00000000
NWA$T_INODEBUF                  000004AC
NWA$T_ITM_ATTR                  00000200
NWA$T_ITM_END                   00000224
NWA$T_ITM_LST                   00000200
NWA$T_ITM_MAXINDX               00000218
NWA$T_ITM_STRING                0000020C
NWA$T_NCBBUF                    0000052C
NWA$T_NODEBUF                   00000169
NWA$T_RCVBUF                    000001A0
NWA$T_SCAN                      00000100
NWA$T_TEMP                      00000120
NWA$T_XLTBUF1                   000002AC
NWA$T_XLTBUF2                   000003AC
NWA$T_XMTBUF                    000003C0
NWA$W_BUILD                     000000D2
NWA$W_DAPBUFSIZ                 000000CA
NWA$W_DIR_OFF                   000000CC
NWA$W_DISPLAY                   000000D0
NWA$W_FIL_OFF                   000000CE
NWA$W_JNLXABJOP                 0000011E
PARTNER                       = 00000030
SYS$ASCTIM                      ********  GX   01
SYS$FAO                         ********   X   01
T_UIC                           00000000 R     01
XAB$B_AID                     = 00000017
XAB$B_ALN                     = 00000009
XAB$B_AOP                     = 00000008
XAB$B_BKZ                     = 00000016
XAB$B_BLN                     = 00000001
XAB$B_DAN                     = 0000000A
XAB$B_DTP                     = 00000013
XAB$B_FLG                     = 00000012
XAB$B_IAN                     = 00000008
XAB$B_LAN                     = 00000009
XAB$B_NUL                     = 00000015
XAB$B_REF                     = 00000017
XAB$B_SIZ                     = 0000002E
XAB$C_CYL                     = 00000001
XAB$C_DATLEN                  = 0000002C
XAB$C_LBN                     = 00000002
XAB$C_STG                     = 00000000
XAB$C_VBN                     = 00000003
XAB$L_ALQ                     = 00000010
XAB$L_KNM                     = 00000038
XAB$L_LOC                     = 0000000C
XAB$L_UIC                     = 00000008
XAB$Q_BDT                     = 00000024
XAB$Q_CDT                     = 00000014
XAB$Q_EDT                     = 0000001C
XAB$Q_RDT                     = 0000000C
XAB$V_CBT                     = 00000005
XAB$V_CHG                     = 00000001
XAB$V_CTG                     = 00000007
XAB$V_DUP                     = 00000000
XAB$V_GRP                     = 00000008
XAB$V_HRD                     = 00000000
XAB$V_NODEL                   = 00000003
XAB$V_NOEXE                   = 00000002
XAB$V_NOREAD                  = 00000000
XAB$V_NOWRITE                 = 00000001
XAB$V_NUL                     = 00000002
XAB$V_ONC                     = 00000001
XAB$V_OWN                     = 00000004
XAB$V_SYS                     = 00000000
XAB$V_WLD                     = 0000000C
XAB$W_DEQ                     = 00000014
XAB$W_DFL                     = 0000001C
XAB$W_GRP                     = 0000000E
XAB$W_IFL                     = 0000001A
XAB$W_MBM                     = 0000000C
XAB$W_POS                     = 0000001E
XAB$W_PRO                     = 00000008
XAB$W_RVN                     = 00000008
XAB$W_VOL                     = 0000000A
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
NF$NETWORK	00000298 (664.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000800 (2048.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.12	00:00:00.97
Command processing	123	00:00:00.61	00:00:06.87
Pass 1	326	00:00:12.82	00:00:27.84
Symbol table sort	0	00:00:01.38	00:00:02.07
Pass 2	125	00:00:02.81	00:00:06.41
Symbol table output	36	00:00:00.30	00:00:00.63
Psect synopsis output	2	00:00:00.03	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	646	00:00:18.07	00:00:44.84

The working set limit was 1500 pages.
66351 bytes (130 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1003 non-local and 35 local symbols.
647 source lines were read in Pass 1, producing 15 object records in Pass 2.
30 pages of virtual memory were used to define 28 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	17
_255DUA28:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	24

1344 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NTOBLDXAB/OBJ=OBJ$:NTOBLDXAB MSRC$:NTOBLDXAB/UPDATE=(ENH$:NTOBLDXAB)+LIB$:RMS/LIB

NT0ACCESS
LIS

NT0CLOSE
LIS

NT0BLDXAB
LIS

NT0CONN
LIS

NT0CREATE
LIS

NT0DAPIO
LIS

NT0DAPCRC
LIS

NT0ACCFIL
LIS

NT0BLKIO
LIS